

### REMARKS

This amendment is submitted in response to the Office Action mailed 6/18/04. Claims in their pending forms are also submitted herein.

1. In Applicant's Response to Office Action dated 4/12/04, Applicant argued that equating a link driver and device driver is inaccurate when link drivers and device drivers are functionally different. The Examiner rejected this argument by stating:

"Specifically, an appropriate device driver corresponding with a device of a plurality of devices should be identified and loaded in the processor [CPU 42 in fig. 3] in order to point-to-point communicate with each other [col. 3, lines 20-22] while the device of the IEEE 1394-1995 standard is supporting a link layer [fig. 1]. Thus, the device driver of the device supporting the link layer includes a link driver."

Applicant respectfully traverses any rejection based on this argument. Device drivers and link drivers are completely separate both in function and structure. If the Examiner wishes to maintain a rejection based on this argument, **Applicant demands proof of Examiner's argument that a device driver includes a link driver.**

Applicant asserts that the Examiner has failed to comprehend the nature of the invention, despite it being plainly stated in both the background of the invention of the present application as well as the claims that define the scope of the invention. Specifically, the present invention is directed to detecting a link driver; receiving capabilities of said link driver; **generating a link driver configuration for said link driver from said capabilities of said driver; and loading said link driver configuration into said link driver.** Examiner's attention is directed to page 2, line 20 – page 3, line 12 of the present application:

"Prior art implementations provide a static link driver which is configured identically for each node 2a, 2b without regard for the type of communication that will be carried out by the node. Thus, link driver 5a is configured the same way as 5b, even though node 2a may carry out different communication than node 2b. The prior art implementation of providing a static configuration for link drivers is not always optimal. For example, in IEEE 1394 communication, nodes may carry out asynchronous and

isochronous communication. In asynchronous communication such as SPB (serial bus protocol), it would be advantageous to have a link device configured for data pumping to provide optimum performance for such asynchronous transfers. On the other hand, in isochronous communication such as AV/C (audio/video control), no advantage is provided if the link device is configured for transferring isochronous data. Thus, the current implementation of providing a static configuration link driver for all LINKS (3a, 3b, for example) is a disadvantage.”

Given this problem statement, and Applicant’s previous attempts to explain to the Examiner the differences between link drivers and devices drivers, the Examiner is again respectfully requested to reconsider the claims. The present invention overcomes the shortcomings of using a **static link driver which is configured identically for each node, without regard for the type of communication that will be carried out by the node.**

Examiner asserts that Shima teaches receiving capabilities of the link driver associated with the device by the link driver accessing the configuration ROM of the device in order to generate an object representing the capabilities of the device.

Applicant respectfully asserts that the Examiner has mischaracterized the cited portion in Shima to support this remark. **Shima, at column 3, lines 38-42, makes no mention of a link driver, or detecting capabilities of a link driver.** Shima, at column 3, lines 38-42 reads:

“During a self-identifying process, after the bus reset, information about the characteristics **of the devices** within the network is received. From this self-identifying information objects representing the devices are generated.”

Applicant respectfully asserts that this portion of Shima is talking about generating objects that represent devices, **not detecting capabilities of a link driver.** If the Examiner wishes to maintain a rejection under 35 USC 102 or 35 USC 103 based on this point, **Applicant demands proof that Shima teaches detecting the capabilities of a link driver.**

2. Shima does not teach disclose or otherwise suggest generating a link driver configuration for the link driver from the capabilities received as claimed in claims 10 and 21.

Examiner asserts that Shima teaches generating a link driver configuration by citing that Shima teaches generating an object representing a device. Examiner has cited Shima at Col. 3, lines 38-42, as teaching receiving capabilities of the link driver associated with the device by the link driver accessing the configuration ROM of the device in order to generate an object represent the capabilities of the device. HOWEVER, col. 3 lines 38-42 in no way support Examiner's argument. Again, Shima at Col. 3, lines 38-42 reads:

"During a self identifying process, after the bus reset, information about the characteristics of the devices within the network is received. From this self-identifying information **objects representing the devices** are generated."

Firstly, Applicant asserts that in no way does the above-cited portion of Shima indicate that a link driver is accessing a configuration ROM. Applicant again asserts that Shima teaches that objects represent devices, and devices are separate functionally and structurally from drivers. **If the Examiner wishes to maintain a rejection based on this point under either 35 USC 102 or 35 USC 103, Applicant demands proof that a device is equivalent to a link driver.**

3. Shima does not teach disclose or otherwise suggest loading the link driver configuration in the link driver as claimed in claims 10 and 21.

Examiner has responded to this argument made by Applicant by stating that Shima teaches that the link driver configuration should be loaded into the link driver to reflect or reconfigure the bus changes or updates [col. 3, lines 22-37; col. 5 lines 41-47] so that the link driver is able to access the device with the capabilities determined [col. 4, lines 1-10]. However, this argument is not supported by the cited portions of Shima.

Shima, at col. 3, lines 22-37 reads:

A reset event signifies to the controlling or monitoring application that the status of the bus and the nodes connected to it has changed. This requires the controlling or

monitoring application to somehow update its information regarding the devices connected to the serial bus network. This can be a significant endeavor. There is currently a lack of efficient apparatus and methods for updating information for nodes on a serial bus network after a bus reset. No solutions for handling the updating of device information after a bus reset is described in any of the IEEE 1394 standards documents.

### SUMMARY OF THE INVENTION

A controlling application utilizes existing handle objects, as appropriate, to reconfigure objects to dynamically enumerate and represent devices coupled to a serial bus network after a bus reset event. Preferably, the serial bus network is an IEEE 1394-1995 serial bus network.

There is no mention of a link driver anywhere in this text. Furthermore, Shima at Col. 3, lines 12-16 clarifies what is meant by the object that is maintained by a controlling or monitoring application:

“Generally, such a controlling or monitoring application maintains a representation or object of each **device**. This object represents the **capabilities of the device**.”

Again, Applicant asserts that a DEVICE is not a LINK DRIVER. This cited portion of Shima does not teach loading a LINK DRIVER generated as claimed in claims 10 and 21 of the present application.

Shima at col. 5, lines 41-47 reads:

“A bus reset occurs when the configuration of the nodes within the IEEE 1394-1995 serial bus network changes. Accordingly, the objects and handles representing devices within the network must somehow be updated after a bus reset to reflect the current state of the nodes coupled to the IEEE 1394-195 serial bus network.”

Again, there is no discussion of a link driver in this cited portion of Shima. An object represents a device. A device is not a link driver. This cited portion does not disclose, teach, nor otherwise suggest loading the link driver configuration in the link driver as claimed in claims 10 and 21.

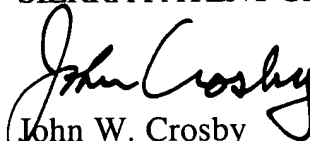
Shima at col. 4, lines 1-10 reads:

“...of maintaining a library of a plurality of available subobjects each representing an available subunit, determining characteristics of the **device**, including resident subunits within the device, retrieving retrieved subobjects from the library corresponding to the resident subunits and assembling the retrieved subobjects into an object representing the functions and characteristics of the device. The method further comprises the step of receiving self identifying information **for the device**. **The characteristics of the device** are determined from the self identifying information.

Again, Applicant asserts that the Examiner has once again mischaracterized the prior art, and this portion in no way teaches, suggests, nor otherwise discloses loading the link driver configuration in the link driver as claimed in claims 10 and 21. Objects represent devices. Devices are not drivers. Objects do not represent link drivers. Shima does not even once discuss a link driver configuration. NO part of Shima relates to loading a link driver configuration into a link driver.

**If the Examiner wishes to maintain a rejection under 35 USC 102 or 35 USC 103 of this claim limitation, Applicant demands proof that the cited prior art teaches, suggests, or otherwise discloses it.**

Respectfully submitted,  
SIERRA PATENT GROUP, LTD.

  
John W. Crosby  
Reg. No. 49,058

Dated: September 22, 2004

Sierra Patent Group, Ltd.  
PO Box 6149  
Stateline, NV 89449  
(775) 586-9500